

# A $W[1]$ -Completeness Result for Generalized Permutation Pattern Matching

Marie-Louise Bruner, Martin Lackner

September 22, 2011

## Abstract

The NP-complete PERMUTATION PATTERN MATCHING problem asks whether a permutation  $P$  (the pattern) can be matched into a permutation  $T$  (the text). A matching is an order-preserving embedding of  $P$  into  $T$ . In the GENERALIZED PERMUTATION PATTERN MATCHING problem one can additionally enforce that certain adjacent elements in the pattern must be mapped to adjacent elements in the text. This paper studies the parameterized complexity of this more general problem. We show  $W[1]$ -completeness with respect to the length of the pattern  $P$ . Under standard complexity theoretic assumptions this implies that no fixed-parameter tractable algorithm can be found for any parameter depending solely on  $P$ .

## 1 Introduction

### 1.1 Permutation patterns

The concept of pattern avoidance (and, closely related, pattern matching) in permutations arose in the late 1960ies when Donald Knuth asked which permutations could be sorted using one stack in an example of his *Fundamental algorithms* [14]. The answer is simple: These are exactly the permutations avoiding the pattern 231 and they are counted by the Catalan numbers. By avoiding a certain pattern the following is meant:

**Definition 1.1.** Let  $P$  be a permutation of length  $k \leq n$  (the pattern). We say that the  $n$ -permutation  $T$  (the text) contains  $P$  as a pattern or that  $P$  can be matched into  $T$  if we can find a subsequence of  $T$  that is order-isomorphic to  $P$ . If there is no such subsequence we say that  $T$  avoids the pattern  $P$ . Matching  $P$  into  $T$  thus consists in finding a monotone map  $\mu : [k] \rightarrow [n]$  so that the sequence  $\mu(P)$ , being defined as  $(\mu(P(i)))_{i \in [k]}$ , is a subsequence of  $T$ .

**Example 1.2.** The text  $T = 53142$  (being a permutation written in one-line representation) contains the pattern 312, since the entries 512 and also 534 form a 312-pattern.  $T$  however avoids the pattern 123 since it contains no increasing subsequence of length three.

Since 1985, when Rodica Simion and Frank Schmidt published the first systematic study of *Restricted Permutations* [18], the area of pattern avoidance in permutations has

become a rapidly growing field of discrete mathematics, more specifically of (enumerative) combinatorics.

There is a far-reaching generalization of pattern matching to so-called *generalized patterns*. These were introduced in [4] and have since then received a lot of attention (see [19] for a survey). In contrast to “classical” pattern matching, some elements of a generalized pattern may be forced to be mapped to adjacent elements in the text. Let us first give an example:

**Example 1.3.** As in the previous example we consider the text  $T = 53142$ . It contains the generalized pattern  $\langle 31 \rangle 2$  as shown by the entries 534. Observe that the subsequence  $\langle 31 \rangle$  of the pattern is mapped to the adjacent elements 53 in the text. This would not have been the case for 512 and hence 512 does not form a  $\langle 31 \rangle 2$ -pattern.  $T$  avoids the pattern  $\langle 312 \rangle$  since it contains no 312-pattern where all three elements are adjacent.

**Definition 1.4.** A generalized pattern  $P$  of length  $k \leq n$  is a permutation of length  $k$  in which parentheses  $\langle \rangle$  have been inserted. These parentheses have to be non-nested, non-overlapping and well-matched, i.e. every parenthesis  $\langle$  has to subsequently be followed by a  $\rangle$ . Also each pair of parentheses has to contain at least two elements. A subsequence of  $P$  surrounded by a pair of parentheses is called adjacent block.

We say that the  $n$ -permutation  $T$  (the text) contains  $P$  as a generalized pattern or that  $P$  can be matched into  $T$  if we can find a subsequence of  $T$  that is order-isomorphic to  $P$  and in which adjacent blocks are mapped to adjacent elements in the text. Matching  $P$  into  $T$  thus consists in finding a monotone map  $\mu : [k] \rightarrow [n]$  so that  $\mu(P)$  is a subsequence of  $T$  and  $\mu(i)$  and  $\mu(j)$  are adjacent whenever  $i$  and  $j$  lie next to each other in an adjacent block.

We remark that this is not the usual notation which uses dashes instead of parentheses. For instance  $\langle 31 \rangle 2$  is usually written as  $31 - 2$ ,  $\langle 312 \rangle$  as  $312$  and  $312$  as  $3 - 1 - 2$ . We prefer to use the parentheses notation since it allows to write classical patterns in the “classical way”.

## 1.2 Computational aspects

This paper takes the viewpoint of computational complexity. Computational aspects of pattern avoidance, in particular the analysis of the PERMUTATION PATTERN MATCHING problem, have so far received far less attention than enumerative questions. This problem is defined as follows:

<p>(GENERALIZED) PERMUTATION PATTERN MATCHING (GPPM/PPM)</p>
<p><i>Instance:</i> A (generalized) pattern <math>P</math> and a permutation <math>T</math> (the text).</p>
<p><i>Question:</i> Is there a matching of <math>P</math> into <math>T</math>?</p>

PPM is also known as SUB-PERMUTATION problem in the literature. In [5] it was shown that the general decision problem is NP-complete. In this paper we focus on the GENERALIZED PERMUTATION PATTERN MATCHING (GPPM). To the best of our knowledge,

so far no work has been done on GPPM. However it is known to be NP-complete, since PPM clearly is a special case.

While knowing about the NP-completeness of this problem, questions regarding its computational complexity remain unanswered. A natural question would be “Is the GENERALIZED PERMUTATION PATTERN MATCHING problem still computationally hard when the pattern is short?”. A framework able to provide an answer is parameterized complexity theory. In contrast to classical complexity theory, parameterized complexity theory studies the complexity of problems with respect to several parameters and not just the input size. Therefore an answer to aforementioned question could be found by performing a parameterized complexity analysis using the length  $k$  of the pattern as a parameter.

A first step towards an algorithm for GPPM is a trivial  $\mathcal{O}(n^k)$  brute-force algorithm. However, even for a moderately long pattern this algorithm is unpractical. In order to solve GPPM efficiently for short patterns, one could hope for a *fixed-parameter tractable* (fpt) algorithm, i.e. having a runtime of  $f(k) \cdot \text{poly}(n)$  with  $f$  being a computable (ideally single-exponential) function.

In this paper we show that under standard complexity theoretic assumptions, no such fpt algorithm exists. This is achieved by showing that GPPM is W[1]-complete with respect to the length of the pattern. This implies, unless  $\text{FPT} = \text{W}[1]$ , that there is no fpt algorithm for GPPM for any parameter that depends solely on the pattern. Furthermore, it follows that an fpt algorithm may only be found if a parameter depending on the text is used.

## 2 Related work

Here we present some relevant work within the complexity analysis of PPM.

- XP-algorithms, i.e. algorithms with runtime  $\mathcal{O}(n^{f(k)})$ , are given in [1, 3].
- Special interest has been shown to the case of *separable* patterns. These are permutations that can be represented with the help of a *separating tree* or, equivalently, permutations containing neither 3142 nor 2413 as a pattern. Note that every permutation avoiding any one of the patterns 132, 231, 213 and 312 is separable. Polynomial time algorithms for the decision and counting problems as well as efficient algorithms for the detection of separable permutations can be found in [3, 5, 13, 16].
- Other cases with restricted patterns have also been studied:
  - In case  $P$  is the identity, PPM consists of looking for an increasing subsequence of length  $k$  in the text which is a special case of the LONGEST INCREASING SUBSEQUENCE problem. This problem can be solved in  $\mathcal{O}(n \log n)$ -time for sequences in general [17] and in  $\mathcal{O}(n \log \log n)$ -time for permutations [7, 15].
  - For all patterns of length four PPM can be solved in  $\mathcal{O}(n \log n)$ -time [3].
  - For the case that both the text and the pattern are 321-avoiding an  $\mathcal{O}(k^2 n^6)$ -time algorithm is presented in [12]. If only the pattern is 321-avoiding an  $\mathcal{O}(kn^{4\sqrt{k}+12})$ -time algorithm was found.

Two other problems related to PPM are:

- The LONGEST COMMON PATTERN problem is to find a longest common pattern between two permutations  $T_1$  and  $T_2$  i.e. a pattern  $P$  of maximal length that can be matched both into  $T_1$  and  $T_2$ . This problem is a generalization of PPM since determining whether the longest common pattern between  $T_1$  and  $T_2$  is  $T_1$  is equivalent to PPM. The LONGEST COMMON PATTERN problem for the case that one of the two permutations  $T_1$  and  $T_2$  is separable has been studied e.g. in [6].
- For a class of permutations  $X$  the LONGEST  $X$ -SUBSEQUENCE (LXS) problem is to identify in a given permutation  $T$  its longest subsequence that is isomorphic to a permutation of  $X$ . In [2] polynomial time algorithms for many classes  $X$  are described, in general however LXS is NP-hard.

As mentioned before, we do not know of any work that has been done on GPPM.

### 3 Preliminaries

We give the relevant definitions of parameterized complexity theory. Details can be found in [9, 11]. In contrast to classical complexity theory, a parameterized complexity analysis studies the runtime of an algorithm with respect to several parameters and not just the input size  $|I|$ . Therefore every parameterized problem is a subset of  $\Sigma^* \times \mathbb{N}$ , where  $\Sigma$  is the input alphabet. An instance of a parameterized problem consequently consists of an input string together with a positive integer  $k$ , the parameter.

**Definition 3.1.** *A parameterized problem  $P$  is fixed-parameter tractable (or in FPT) if there is a computable function  $f$  and a polynomial  $p$  such that there is an algorithm solving  $P$  in time  $f(k) \cdot p(|I|)$ . Such an algorithm is called fixed-parameter tractable as well.*

We continue by defining parameterized reductions.

**Definition 3.2.** *Let  $L_1, L_2 \subseteq \Sigma^* \times \mathbb{N}$  be two parameterized problems. An fpt-reduction from  $L_1$  to  $L_2$  is a mapping  $R : \Sigma^* \times \mathbb{N} \rightarrow \Sigma^* \times \mathbb{N}$  such that*

1.  $(I, k) \in L_1$  iff  $R(I, k) \in L_2$ .
2.  $R$  is fixed-parameter tractable.
3. There is a computable function  $g$  such that for  $R(I, k) = (I', k')$ ,  $k' \leq g(k)$  holds.

FPT is the parameterized equivalent of PTIME. Other important complexity classes in the framework of parameterized complexity are those in the W-hierarchy,  $W[1] \subseteq W[2] \subseteq \dots$ . For our purpose only the class  $W[1]$  is relevant. It is conjectured (and widely believed) that  $W[1] \neq \text{FPT}$  (see e.g. [8]). Therefore showing  $W[1]$ -hardness can be considered as evidence that the problem is not fixed-parameter tractable.

**Definition 3.3.** *The class  $W[1]$  is defined as the class of all problems that are fpt-reducible to the following problem.*

**TURING MACHINE ACCEPTANCE**

*Instance:* A nondeterministic Turing machine with its transition table,  
an input word  $x$  and a positive integer  $k$ .  
*Parameter:*  $k$   
*Question:* Does the Turing machine accept the input  $x$  in at most  $k$  steps?

**Definition 3.4.** A parameterized problem  $P$  is in  $\text{XP}$  if there is a computable function  $f$  such that there is an algorithm solving  $P$  in time  $\mathcal{O}(|I|^{f(k)})$ .

All the aforementioned classes are closed under fpt-reductions. The following relations between these complexity classes are known:

$$\text{FPT} \subseteq \text{W}[1] \subseteq \text{W}[2] \subseteq \dots \subseteq \text{XP}$$

Finally,  $[k]$  denotes the set  $\{1, \dots, k\}$ .

## 4 The $\text{W}[1]$ -completeness result

**Theorem 4.1.** GPPM is  $\text{W}[1]$ -complete with respect to length of the pattern.

*Proof.* We show  $\text{W}[1]$ -hardness by giving an fpt-reduction from the following problem to GPPM:

**INDEPENDENT SET**

*Instance:* A graph  $G = (V, E)$  and a positive integer  $k$ .  
*Parameter:*  $k$   
*Question:* Is there a subset  $S \subseteq V$  of size  $k$ , so that the induced subgraph  $G[S]$  contains no edges?

Let  $(G, k)$  be an INDEPENDENT SET instance, where  $V = \{v_1, v_2, \dots, v_l\}$  is the set of vertices and  $E = \{e_1, e_2, \dots, e_m\}$  the set of edges. We are going to construct a GPPM instance  $(P, T)$ . We shall first construct a pair  $(P', T')$ .  $P'$  is a generalized multiset pattern, i.e. a generalized pattern in which elements may occur more than once.  $T'$  is a permutation on a multiset. Applying Definition 1.4 to permutations on multisets means that in a matching repeated elements in the pattern have to be mapped to repeated elements in the text. Afterwards we deal with the repeated elements in order to create a generalized pattern and a permutation on ordinary sets and hereby obtain  $(P, T)$ .

Both the pattern and the text consist of a substring coding vertices ( $\dot{P}$  resp.  $\dot{T}$ ) and a substring coding edges ( $\bar{P}$  resp.  $\bar{T}$ ). In between these two substrings we place a *separator block* of constant length  $c$  to ensure that  $\dot{P}$  is matched into  $\dot{T}$  and  $\bar{P}$  into  $\bar{T}$ . For the moment, we will simply write  $\|$  for this separator block, indicating that the separator block in the pattern has to be mapped to the one in the text. Its construction shall be described later on.

We define the pattern to be

$$P' = \dot{P} \| \bar{P} = 123 \dots k \| \langle 121 \rangle \langle 131 \rangle \dots \langle 1k1 \rangle \langle 232 \rangle \dots \langle 2k2 \rangle \dots \langle (k-1)k(k-1) \rangle$$

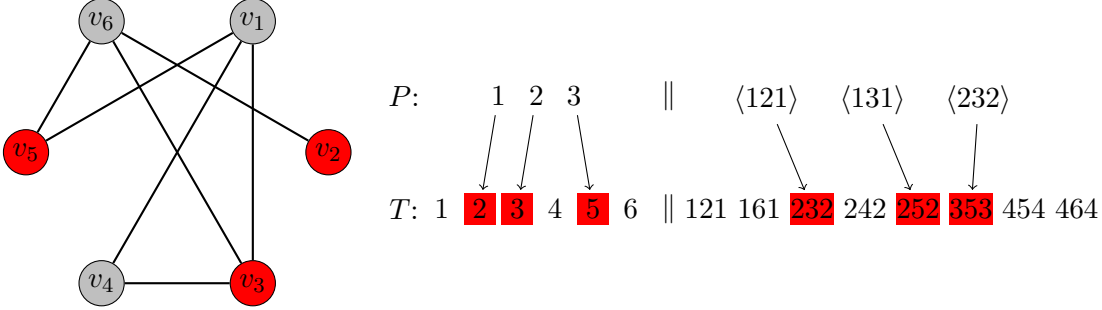


Figure 1: An example for the reduction of an INDEPENDENT SET instance to a PPM instance.

$\dot{P}$  corresponds to a list of (indices of)  $k$  vertices.  $\bar{P}$  represents all possible edges between the  $k$  vertices (in lexicographic order). An edge from  $i$  to  $j$  is encoded by  $\langle iji \rangle$ . Since the complete graph on  $k$  vertices has  $k(k-1)/2$  edges the total length of  $P$  is equal to  $k + c + 3k(k-1)/2 = c + (3k^2 - k)/2$ .

For the text  $T' = \dot{T} \parallel \bar{T}$  we proceed similarly.  $\dot{T}$  is a list of the (indices of the)  $l$  vertices of  $G$ .  $\bar{T}$  represents all edges *not occurring* in  $G$  listed in lexicographic order. An edge  $\{i, j\}$  is again encoded by  $iji$ . Let us give an example:

**Example.** Let  $l = 6$  and  $k = 3$ . Then the pattern is given by

$$P' = 123 \parallel \langle 121 \rangle \langle 131 \rangle \langle 232 \rangle$$

Consider for instance the graph  $G$  with six vertices  $v_1, \dots, v_6$  and edge-set

$$\{\{1, 3\}, \{1, 4\}, \{1, 5\}, \{2, 6\}, \{3, 4\}, \{3, 6\}, \{5, 6\}\}$$

represented in Figure 1 (we write  $\{i, j\}$  instead of  $\{v_i, v_j\}$ ). Then the eight edges not appearing in  $G$  are

$$\{\{1, 2\}, \{1, 6\}, \{2, 3\}, \{2, 4\}, \{2, 5\}, \{3, 5\}, \{4, 5\}, \{4, 6\}\}.$$

The text is thus given by:

$$T' = 123456 \parallel 121 \ 161 \ 232 \ 242 \ 252 \ 353 \ 454 \ 464.$$

Since there are at most  $l(l-1)/2$  edges not appearing in  $G$ , the length of  $T'$  is at the most  $c + (3l^2 - l)/2$ .

**Claim 1.** An independent set of size  $k$  can be found in  $G$  if and only if there is a simultaneous matching of  $\dot{P}$  into  $\dot{T}$  and of  $\bar{P}$  into  $\bar{T}$ .

**Example** (continuation). In our example  $\{v_2, v_3, v_5\}$  is an independent set of size three. Indeed, the pattern  $P'$  can be matched into  $T'$  as can be seen by matching the elements 1, 2 and 3 onto 2, 3 and 5 respectively. See again Figure 1 where the involved vertices respectively elements of the text have been marked in red.

*Proof of Claim 1.* A matching of  $\dot{P}$  into  $\dot{T}$  corresponds to a selection of  $k$  vertices amongst the  $l$  vertices of  $G$ . If it is possible to additionally match  $\bar{P}$  into  $\bar{T}$  this means that there are no edges between the selected vertices. Observe that a matching of the generalized pattern  $\bar{P}$  into  $\bar{T}$  consists of mapping each edge between selected vertices to an edge not appearing in  $G$ . This is because every adjacent triple of the form  $iji$  (with  $i < j$ ) in  $\bar{T}$  corresponds to one of the non-edges of the graph. The selected  $k$  vertices thus form an independent set in  $G$ . Conversely, if for every possible matching of  $\dot{P}$  into  $\dot{T}$  defined by a monotone map  $\mu : [k] \rightarrow [l]$  some  $\langle xyx \rangle$  in  $\bar{P}$  cannot be matched into  $\bar{T}$ , this means that  $\{\mu(x), \mu(y)\}$  is one of the edges of  $G$  since  $\mu(x)\mu(y)\mu(x)$  does not appear as adjacent elements in  $\bar{T}$ . Thus, for every selection of  $k$  vertices there will always be at least one edge connecting two of them and therefore there is no independent set of size  $k$  in  $G$ .  $\square$

In order to get rid of repeated elements, we identify every variable with a real interval: 1 corresponds to the interval  $[1, 1.9]$ , 2 to  $[2, 2.9]$  and so on until finally  $k$  corresponds to  $[k, k + 0.9]$  (resp.  $l$  to  $[l, l + 0.9]$ ). In  $\dot{P}$  and  $\dot{T}$  we shall therefore replace every element  $j$  by the pair of elements  $(j + 0.9, j)$  (in this order). The occurrences of  $j$  in  $\bar{P}$  (resp.  $\bar{T}$ ) shall then successively be replaced by real numbers in the interval  $[j, j + 0.9]$ . For every  $j$ , these values are chosen one after the other (from left to right), always picking a real number that is larger than all the previously chosen ones in the interval  $[j, j + 0.9]$ . Note that this construction changes the length of the pattern (resp. the text) by  $k$  (resp.  $l$ ) since the length of  $\dot{P}$  (resp.  $\dot{T}$ ) is doubled. We now have  $|P| = c + (3k^2 + k)/2$  and  $|T| \leq c + (3l^2 + l)/2$ .

Observe the following: The obtained sequence is not a permutation in the classical sense since it consists of real numbers. However, by replacing the smallest number by 1, the second smallest by 2 and so on, we do obtain an ordinary permutation. This defines  $P$  and  $T$  (except for the separator block).

**Example** (continuation). Getting rid of repetitions in the pattern of the above example could for instance be done in the following way:

$$P = 1.9 \ 1 \ 2.9 \ 2 \ 3.9 \ 3 \parallel \langle 1.1 \ 2.1 \ 1.2 \rangle \langle 1.3 \ 3.1 \ 1.4 \rangle \langle 2.2 \ 3.2 \ 2.3 \rangle$$

This permutation of real numbers is order-isomorphic to the following ordinary permutation:

$$P = 61B7FC \parallel \langle 283 \rangle \langle 4D5 \rangle \langle 9EA \rangle$$

For increased legibility, we use letters A, B, C,... with lexicographic order for numbers larger than 9, i.e. A = 10, B = 11, and so on.

**Claim 2.**  $P$  can be matched into  $T$  iff  $P'$  can be matched into  $T'$ .

*Proof of Claim 2.* Suppose that  $P'$  can be matched into  $T'$ . When matching  $P$  into  $T$ , we have to make sure that elements in  $P$  that were copies of some repeated element in  $P'$  may still be mapped to elements in  $T$  that were copies themselves in  $T'$ . Indeed this is possible since we have chosen the real numbers replacing repeated elements in increasing order. If  $i$  in  $P'$  was matched to  $j$  in  $T'$ , then the pair  $(i + 0.9, i)$  in  $P$  may be matched to the pair  $(j + 0.9, j)$  in  $T$  and the increasing sequence of elements in the interval  $[i, i + 0.9]$  may be matched into the increasing sequence of elements in the interval  $[j, j + 0.9]$ .

Now suppose that  $P$  can be matched into  $T$ . In order to prove that this implies that  $P'$  can be matched into  $T'$ , we merely need to show that elements in  $P$  that were copies of some repeated element in  $P'$  have to be mapped to elements in  $T$  that were copies themselves in  $T'$ . Then returning to repeated elements clearly preserves the matching. Firstly, it is clear that a pair of consecutive elements  $i + 0.9$  and  $i$  in  $\dot{P}$  has to be matched to some pair of consecutive elements  $j + 0.9$  and  $j$  in  $\dot{T}$ , since  $j$  is the only element smaller than  $j + 0.9$  and appearing to its right. Thus intervals are matched to intervals. Secondly, an element  $x$  in  $P$  for which it holds that  $i < x < i + 0.9$  must be matched to an element  $y$  in  $T$  for which it holds that  $j < y < j + 0.9$ . Thus copies of an element are still matched to copies of some other element.

Finally, replacing real numbers by integers clearly does not change the permutations in any relevant way.  $\square$

We now have to implement the separator block in order to ensure that the two substrings  $\dot{P}$  and  $\bar{P}$  of the pattern are matched into the corresponding substrings  $\dot{T}$  resp.  $\bar{T}$  in  $T$ . In  $P$  we proceed in the following way. Let us denote its largest element by  $P_{\max}$  (this is the next-to-last element of  $\dot{P}$  and was denoted by  $(k + 0.9)$  in  $P'$ ). Then we define the separator block to be the generalized pattern  $\langle P_{\max} + 3, P_{\max} + 2, P_{\max} + 1, P_{\max} + 4 \rangle$ . This leads to  $P = \dot{P} \langle P_{\max} + 3, P_{\max} + 2, P_{\max} + 1, P_{\max} + 4 \rangle \bar{P}$ . For  $T$  we proceed similarly and obtain  $T = \dot{T} (T_{\max} + 3) (T_{\max} + 2) (T_{\max} + 1) (T_{\max} + 4) \bar{T}$ . We thus have  $c = 4$ , i.e. the separator block has length 4.

**Example** (continuation). The largest element of  $P$  is  $F = 15$ , thus we will use the following three elements as separator block: G, H, I and J. Inserting these elements as described above leads to:

$$P = 61B7FC \langle IHGJ \rangle \langle 283 \rangle \langle 4D5 \rangle \langle 9EA \rangle$$

**Claim 3.** *Constructing the separator block in the described way guarantees that in a matching of  $P$  into  $T$ ,  $\dot{P}$  is matched into  $\dot{T}$  and  $\bar{P}$  is matched into  $\bar{T}$ .*

*Proof of Claim 3.* The only  $\langle 3214 \rangle$ -pattern that can be found in  $T$ , i.e. the only decreasing subsequence consisting of three adjacent elements followed by a larger element, is formed by the elements  $(T_{\max} + 3)(T_{\max} + 2)(T_{\max} + 1)(T_{\max} + 4)$ . Thus the separator block in  $P$  must be mapped to the one in  $T$ . Consequently the elements lying to its left (these are exactly the elements of  $\dot{P}$ ) must be matched into the elements lying to the left of the separator block in  $T$ , i.e. into  $\dot{T}$ . For the same reason  $\bar{P}$  must be matched into  $\bar{T}$ .  $\square$



This finally yields that  $(G, k)$  is a YES-instance of INDEPENDENT SET if and only if  $(P, T)$  is a YES-instance of GPPM. It remains to show that this reduction can be done in fpt-time. We have already remarked that the length of the pattern is equal to  $4 + (3k^2 + k)/2$  and the length of the text is at the most  $4 + (3l^2 + l)/2$ . Thus  $|P| = \mathcal{O}(k^2)$  and  $|T| = \mathcal{O}(l^2)$ .

For showing membership we encode GPPM as a model checking problem of an existential first order formula.  $W[1]$ -membership is then a consequence of the fact that the following problem is  $W[1]$ -complete [10].

**EXISTENTIAL FIRST-ORDER MODEL CHECKING**

*Instance:* A structure  $\mathcal{A}$  and an existential first-order formula  $\varphi$   
*Parameter:*  $|\varphi|$   
*Question:* Is  $\mathcal{A}$  a model for  $\varphi$ ?

Let  $k = |P|$ . We compute a structure  $\mathcal{A} = (A, <, T_<, S)$ , where the domain set  $A = \{1, \dots, n\}$  represents indices in the text.  $T_<$  is a binary relation where  $T_<(x, y)$  is true iff the element on the  $x$ -th position in  $T$  is less than the element on position  $y$ .  $S$  is a binary relation where  $S(x, y)$  is true iff  $y$  is the successor of  $x$ , i.e.  $x + 1 = y$ .  $T_<$ ,  $S$  and  $<$  can be computed in polynomial time. The formula  $\varphi$  we want to check is

$$\varphi = \exists x_1 \dots \exists x_k \quad x_1 < x_2 \wedge x_2 < x_3 \wedge \dots \wedge x_{k-1} < x_k \wedge$$

$$\bigwedge_{\substack{P(i) < P(j) \\ \text{for } i, j \in [k]}} T_<(x_i, x_j) \wedge \bigwedge_{\substack{P(i) > P(j) \\ \text{for } i, j \in [k]}} \neg T_<(x_i, x_j) \wedge \bigwedge_{\substack{AdjToRight(i) \\ \text{for } i \in [k-1]}} S(x_i, x_{i+1}),$$

where  $AdjToRight(i)$  is true iff  $P(i)P(i+1)$  is contained in an adjacent block. Observe that the length of  $\varphi$  is in  $\mathcal{O}(k^2)$ . The correctness of the reduction follows directly from Definition 1.4. Indeed, the fact that  $P$  can be matched into  $T$  means that indices  $x_1, \dots, x_k$  where  $x_1 < \dots < x_k$  can be found so that  $P(i) < P(j)$  iff  $T(x_i) < T(x_j)$  and so that  $x_i + 1 = x_{i+1}$  holds whenever  $AdjToRight(i)$  is true.  $\square$

*Remark 4.2.* Since in the fpt-reduction the length of the pattern can be bounded by a polynomial in the size of  $G$ , this is also a polynomial time reduction. Therefore the proof of Theorem 4.1 can also be seen as an alternative way of showing NP-hardness for GPPM. This also follows from the NP-hardness of the less general PPM.

## 5 Conclusion

We have shown that the GENERALIZED PERMUTATION PATTERN MATCHING problem is  $W[1]$ -complete with respect to the length of the pattern. This implies that under standard complexity theoretic assumptions there is no fpt-algorithm for this problem with respect to the length of the pattern. Furthermore, this also yields that *no* parameter that is a function of the pattern can yield fixed-parameter tractability. In order to obtain fpt results, we plan to study parameters concerning the text. The question whether PPM is also  $W[1]$ -hard remains open.

## 6 Acknowledgements

We would like to thank Marek Cygan, Marcin Pilipczuk and Ondřej Suchý for pointing out a flaw in the proof of a previous version of this paper.

## References

- [1] Shlomo Ahal and Yuri Rabinovich. On complexity of the subpattern problem. *SIAM J. Discrete Math.*, 22(2):629–649, 2008.
- [2] M. H. Albert, R. E.L. Aldred, M. D. Atkinson, H. P. van Ditmarsch, B. D. Handley, C. C. Handley, and J. Opatrny. Longest subsequences in permutations. *Australasian Journal of Combinatorics*, 28:225–238, 2003.
- [3] Michael H. Albert, Robert E. L. Aldred, Mike D. Atkinson, and Derek A. Holton. Algorithms for pattern involvement in permutations. In *ISAAC*, volume 2223 of *Lecture Notes in Computer Science*, pages 355–366. Springer, 2001.
- [4] E. Babson and E. Steingrímsson. Generalized permutation patterns and a classification of the mahonian statistics. *Sém. Lothar. Combin.*, 44:117, 2000.
- [5] Prosenjit Bose, Jonathan F. Buss, and Anna Lubiw. Pattern matching for permutations. In *WADS*, volume 709 of *Lecture Notes in Computer Science*, pages 200–209. Springer, 1993.
- [6] Mathilde Bouvel and Dominique Rossin. The longest common pattern problem for two permutations. *Pure Mathematics and Applications*, 17(1-2):5569, 2006.
- [7] Maw-Shang Chang and Fu-Hsing Wang. Efficient algorithms for the maximum weight clique and maximum weight independent set problems on permutation graphs. *Inf. Process. Lett.*, 43(6):293–295, 1992.
- [8] Stefan S. Dantchev, Barnaby Martin, and Stefan Szeider. Parameterized proof complexity. *Computational Complexity*, 20(1):51–85, 2011.
- [9] R.G. Downey and M.R. Fellows. *Parameterized Complexity*. Springer, 1999.
- [10] Jörg Flum and Martin Grohe. Model-checking problems as a basis for parameterized intractability. *Logical Methods in Computer Science*, 1(1), 2005.
- [11] Jörg Flum and Martin Grohe. *Parameterized complexity theory*. Springer, 2006.
- [12] Sylvain Guillemot and Stéphane Vialette. Pattern matching for 321-avoiding permutations. In *ISAAC*, volume 5878 of *Lecture Notes in Computer Science*, pages 1064–1073. Springer, 2009.
- [13] Louis Ibarra. Finding pattern matchings for permutations. *Inf. Process. Lett.*, 61(6):293–295, 1997.

- [14] Donald E. Knuth. *The Art of Computer Programming, Volume I: Fundamental Algorithms*. Addison-Wesley, 1968.
- [15] Erkki Mäkinen. On the longest upsequence problem for permutations. *International journal of computer mathematics*, 77(1):45–53, 2001.
- [16] Sanjeev Saxena and V. Yugandhar. Parallel algorithms for separable permutations. *Discrete Applied Mathematics*, 146(3):343–364, 2005.
- [17] Craige Schensted. Longest increasing and decreasing subsequences. *Classic Papers in Combinatorics*, pages 299–311, 1987.
- [18] Rodica Simion and Frank W. Schmidt. Restricted permutations. *European Journal of Combinatorics*, 6:383–406, 1985.
- [19] E. Steingrímsson. Generalized permutation patternsa short survey. *Permutation Patterns*, 376:137–152.